Statistical Techniques in Robotics (16-831, S21) Lecture #19 (Wednesday, April 14)

Model-Free Value Prediction and Control

Lecturer: Kris Kitani *Scribes: Akshay Dharmavaram, Zongyue Zhao*

1 Review

1.1 Introduction

In the previous lectures, we discussed various methods for on-policy model-free policy evaluation. Specifically, we covered the Monte Carlo (MC) method, one-step Temporal Difference (TD(0)), N-step TD, and the offline λ-Return method. We also proved that the TD error in λ-Return can be expressed as the weighted sum of a series of one step TD errors. In this lecture, we used this proposition to extend λ-Return to the online TD(λ), by introducing the concept of eligibility trace. Moreover, we discussed importance sampling and off-policy prediction, where the target policy - which is corresponding to the value function - is different from the sampling policy generating the trajectories.

In the following sections, we first provide a quick recap on the on-policy methods, which are characterized by the return estimate, then move on to the content of this lecture in details.

1.2 Monte Carlo

In general, the term "Monte Carlo" is used to describe a series of estimation methods that incorporate significant randomness [1]. In the context of reinforcement learning, MC methods refer to methods based on the average of complete sample returns. The return estimate for a single update is thus defined as:

$$G^{(t)} = \sum_{i=t}^{T} r^(i) \tag{1}$$

Here, because the sample return is not available until the episode is terminated, our algorithm is only incremental with respect to episodes, instead of actions. The benefit of doing so is that each estimate $G^{(t)}$ is a direct unbiased estimate of $V^{\pi}(s)$, and the average of these targets is still an unbiased estimator of the value function. On the other hand, the variance of such return-based target is high, especially when the sequence is long.

1.3 Temporal Difference

In order to address the shortcomings of MC methods, we introduced bootstrapping to incrementally update values after every action. Specifically, TD methods use a series of discounted rewards, which is a fragment of the total return, and the current estimate of the value function in place of the

missing terms in the return, as the update target:

$$G^{(t)}(N) = \sum_{i=t}^{t+N-1} \left(\gamma^{i-t} r^{(i)}\right) + \gamma^N V(s^{(t+N)}) \tag{2}$$

and the special case where $N = 1$ is called TD(0):

$$G_0^t \stackrel{\text{def}}{=} G^{(t)}(1) = r^t + \gamma V(s^{(t+1)}) \tag{3}$$

Here, because we used the discounted value function in place of the missing terms in the true sampled return, each update is biased, and the more missing items (corresponding to a low N) the higher bias. However, these methods make it possible to update the value function incrementally with respect to actions, and thus the variance is lower.

1.4 Offline λ-Return

Finding the optimal N in $TD(N)$ methods, i.e., finding the optimal point in the bias-variance trade-off, could be tricky. One method is to combine all temporal difference estimates by assigning higher weights to closer estimates, which is called λ-return:

$$G_\lambda^{(t)} = (1-\lambda) \sum_{n=1}^{\infty} \lambda^{n-1} G^{(t)}(n) \tag{4}$$

if the horizon of the task is infinite. For truly episodic tasks, the λ-return becomes:

$$G_\lambda^{(t)} = (1-\lambda) \sum_{n=1}^{T-t-1} \left(\lambda^{n-1} G^{(t)}(n)\right) + \lambda^{T-t-1} G^{(t)}(T-t-1) \tag{5}$$

Here, if the decay factor λ is close to 0, the algorithm heavily punishes estimates with long sample reward sequence, and the algorithm behaves like TD(0). On the other hand, when the decay factor is close to 1, the algorithm poses less punishment towards long estimates. In the extreme case where no decay is used ($\lambda = 1$), the algorithm is simply Monte Carlo.

Nonetheless, the λ-return method requires all TD updates up until the one extending to the end of the episode, thus the algorithm is offline (incremental w.r.t. episodes). In order to make λ-return an online algorithm, we need to express the corresponding TD error as a sum of weighted 1-step TD errors:

$$\Delta V\left(s^{(t)}\right) = G_\lambda^{(t)} - V\left(s^{(t)}\right) = \sum_{i=0}^{\infty} (\gamma\lambda)^i \delta^{(t+i)} \tag{6}$$

where $\delta^{(\tau)} = r^{(\tau)} + \gamma V\left(s^{(\tau+1)}\right) - V\left(s^{(\tau)}\right)$ is the one-step TD error at τ.

2 Online TD (λ)

2.1 Eligibility Trace

Consider the update rule of λ-return. If we use the offline target shown in Equation 4 or 5, we are essentially looking forward in time to all future rewards, which are encoded in the various n-step

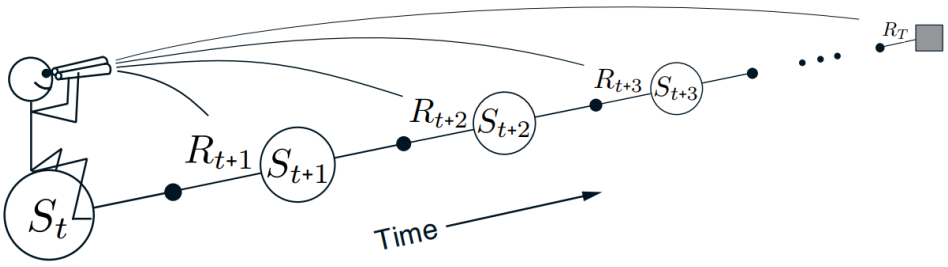

Figure 1: The Forward View [1].

TD errors, and try to find a way to optimally combine them. As shown in Figure 1, this is called the forward view of a learning algorithm. The problem of doing so is that it requires information of the unexplored parts of the trajectory when iterating through each time stamp, thus cannot be directly implemented online.

One method of extending the offline λ-return algorithm is to introduce the concept of eligibility trace. Consider Equation 6, where we decomposed the TD error of the offline λ-return algorithm into the sum of a series of 1-step TD errors:

$$\Delta V\left(s^{(t)}\right) = \sum_{i=0}^{\infty} (\gamma\lambda)^i \delta^{(t+i)} \tag{7}$$

Let us consider such an update on a specific state: suppose at timestamp k, $s^{(k)} = s$, then Equation 7 can be written as:

$$\Delta V\left(s^{(k)} = s\right) = \sum_{i=0}^{\infty} (\gamma\lambda)^i \delta^{(k+i)} = \sum_{t=k}^{\infty} (\gamma\lambda)^{t-k} \delta^{(t)} \tag{8}$$

by shifting superscript notations.

The summation here begins from $t = k$, the current occurrence of s. For a consistent representation on the entire time horizon, we introduce zero-padding for time stamps prior to $t = k$, and the TD target now becomes:

$$\Delta V\left(s^{(k)} = s\right) = \sum_{t=k}^{\infty} (\gamma\lambda)^{t-k} \delta^{(t)} = \sum_{t=1}^{\infty} z^{(t)}(s)\delta^{(t)}, \text{ where } z^{(t)}(s) = \begin{cases} 0 & t < k \\ (\gamma\lambda)^{t-k} & t \geq k \end{cases} \tag{9}$$

The weighting function $z^{(t)}(s)$ is called the eligibility trace. At this stage, we can use it to rewrite the λ-return update rule:

$$V(s^{(t)}) \leftarrow V(s^{(t)}) + \alpha \Delta V\left(s^{(k)} = s\right) \tag{10}$$

to:

$$V(s) \leftarrow V(s) + \sum_{t=1}^{\infty} z^{(t)}(s)\delta^{(t)} \tag{11}$$

Note that the update rule 11 is still offline when calculating explicitly, because it demands information from the entire horizon $t = 1 : \infty$. However, because of the structure of the eligibility

trace $z^{(t)}$, we can adopt a recursive, online implementation by iterating through all states at each timestamp:

$$\forall t, \forall s, z(s) \leftarrow \gamma\lambda z(s) + \mathbb{1}[s^{(t)} = s] \tag{12}$$

The reason we could not do so with the raw representation of $\Delta V\left(s^{(t)}\right) = G_\lambda^{(t)} - V\left(s^{(t)}\right)$ is that $G_\lambda^{(t)}$ spans in the entire episode horizon but $V\left(s^{(t)}\right)$ is a given value at the current timestamp, thus recursion is not available.

The full online algorithm is called $TD(\lambda)$, as shown in Algorithm 1.

Algorithm 1 TD(λ-Prediction$(\pi, \alpha, \lambda, \gamma, s^{(0)})$

for $s \in \mathcal{S}$ **do**
  $z(s) \leftarrow 0$ ▷ Initialize eligibility trace to zero vector.
end for
for $t = 0, ..., T$ **do**
  $\{s^{(t+1)}, a^{(t)}, r^{(t)}\} \sim \mathcal{E}|\pi, s^{(t)}$ ▷ Select and execute action, receive reward
  $\delta \leftarrow r^{(t)} + \gamma V\left(s^{(t+1)}\right) - V\left(s^{(t)}\right)$ ▷ 1-step TD error.
  for $s \in \mathcal{S}$ **do**
    $z(s) \leftarrow \gamma\lambda z(s) + \mathbb{1}\left[s^{(t)} = s\right]$ ▷ Update eligibility trace.
    $V(s) \leftarrow V(s) + \alpha\delta z(s)$ ▷ Update value function
  end for
end for
return $V(s)$

Here, the online TD(λ) algorithm adopts the backward view (note how the value function is updated for any state visited in the past, in a single timestamp, whereas in the forward view, only one - the current - state is updated in a single timestamp). Furthermore, an intuitive demonstration of TD(λ) is shown in Figure 2: each update only depends on the current TD error combined with the eligibility traces summarizing past events.

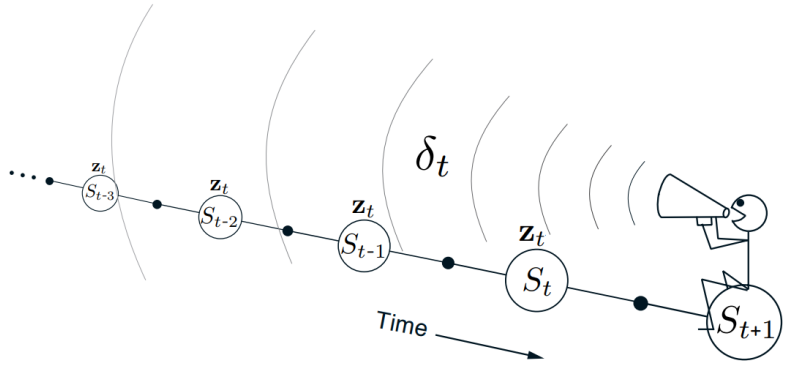

Figure 2: Backward View of TD(λ) [1]

The decay factor λ controls the behavior of Algorithm 1. When $\lambda = 0$, which means $z(s)$ carries no information into the future, the algorithm simply yields the 1-step TD prediction (hence it is called $TD(0)$). On the other hand, when $\lambda = 1$, there is no decay on the eligibility trace, and the algorithm becomes a variant of the discounted Monte Carlo Algorithm.

3 Off-Policy Model-free Prediction

3.1 Off-Policy vs On-Policy

Up until now, we have discussed the various on-policy model-free methods, and from here on we will be transitioning to off-policy methods. We will be discussing 2 methods, namely Importance Sampling for Off-policy MC Prediction (IS-MC) and Importance Sampling for Off-policy Temporal Differencing (IS-TD).

Algorithms	Monte Carlo	Temporal Differencing
On-Policy	Monte Carlo	1) N-step TD 2) TD(lambda)
Off-Policy	Importance Sampling Monte-Carlo	Importance Sampling Temporal Differencing

Table 1: Model-free Prediction Methods Covered

Off-policy methods mainly defer from on-policy methods in their sampling steps. In an off-policy algorithm, we sample our state, action , and reward values from a distinct behavioral policy (μ), which need not be our current target policy (π) that conditionally defines our value-function. Off-policy methods are useful in scenarios such as:

- Learning a policy using the experience of other policies.
- Increasing the sample efficiency of traditional on-policy objectives.
- Evaluate multiple target policies using the experiences from a single behavioral policy.
- Evaluating a policy using different past experiences, defined by different behavioral policies.

As the samples in the off-policy setting are not from the target policy, if we keep the rest of the on-policy algorithm the same, we will end up converging upon the value-function that is conditioned over the behavioral policy (μ) rather than the target policy (π). Thus, we will naturally have to change the value-function update equations to take into account the switch in the sampling procedure. We use the method of imporatnce sampling, to modify the value-function update steps, in order to obtain the correct value-function, which is conditioned on the target policy (π), instead of the behavioral policy (μ).

In the upcoming sections, we will derive the the update equation involving importance sampling for the IS-MC and IS-TD methods, and finally, we will present the pseudocode for all the model-free prediction algorithms discussed till now.

3.2 Importance Sampling for Off-policy MC Prediction

In this section, we introduce the concept of importance sampling for off-policy MC prediction, and show how to update the value-function update steps, to converge upon the correct value-function. Importance sampling is a technique that helps us estimate the values of the value-function of the target policy, when given samples from a different a behavioral policy.

We will derive the importance sampling based update equation below. We start by expanding our value-function, based on our pre-defined definitions.

$$V^{\pi}(s) = \mathbb{E}_{\pi}\left[G^{(t)} \mid s^{(t)} = s\right] \tag{13}$$

$$= \sum_{s^{t+1:T}, a^{t:T-1}} \prod_{k=t}^{T-1} \pi\left(a^{(k)} \mid s^{(k)}\right) p\left(s^{(k+1)} \mid s^{(k)}, a^{(k)}\right) G^{(t)} \tag{14}$$

We can see that our current rollouts are conditional sampled from the target policy, π. Thus, we will introduce a trick that will help us transition the sampling process to one sampled by the behavioral policy, μ. We multiply the numerator and the denominator with the corresponding sampled sequence from the behavioral policy, μ. This does not affect the mathematical integrity of our update, as these values cancel out each other mathematically.

Next, we rearrange the numerator and the denominator to get a sampled sequence that is defined by our behavioral policy.

$$V^{\pi}(s) = \sum_{s^{t+1:T}, a^{t:T-1}} \frac{\prod_{k=t}^{T-1} \mu\left(a^{(k)} \mid s^{(k)}\right) p\left(s^{(k+1)} \mid s^{(k)}, a^{(k)}\right)}{\prod_{k=t}^{T-1} \mu\left(a^{(k)} \mid s^{(k)}\right) p\left(s^{(k+1)} \mid s^{(k)}, a^{(k)}\right)} \prod_{k=t}^{T-1} \pi\left(a^{(k)} \mid s^{(k)}\right) p\left(s^{(k+1)} \mid s^{(k)}, a^{(k)}\right) G^{(t)} \tag{15}$$

$$= \sum_{s^{t+1:T}, a^{t:T-1}} \prod_{k=t}^{T-1} \mu\left(a^{(k)} \mid s^{(k)}\right) p\left(s^{(k+1)} \mid s^{(k)}, a^{(k)}\right) \frac{\prod_{k=t}^{T-1} \pi\left(a^{(k)} \mid s^{(k)}\right) p\left(s^{(k+1)} \mid s^{(k)}, a^{(k)}\right)}{\prod_{k=t}^{T-1} \mu\left(a^{(k)} \mid s^{(k)}\right) p\left(s^{(k+1)} \mid s^{(k)}, a^{(k)}\right)} G^{(t)} \tag{16}$$

$$= \sum_{s^{t+1:T}, a^{t:T-1}} \prod_{k=t}^{T-1} \mu\left(a^{(k)} \mid s^{(k)}\right) p\left(s^{(k+1)} \mid s^{(k)}, a^{(k)}\right) \frac{\prod_{k=t}^{T-1} \pi\left(a^{(k)} \mid s^{(k)}\right)}{\prod_{k=t}^{T-1} \mu\left(a^{(k)} \mid s^{(k)}\right)} G^{(t)} \tag{17}$$

We finally reformulate the equation as an expectation and define a new importance sampling ratio $\frac{\prod_{k=t}^{T-1} \pi\left(a^{(k)} \mid s^{(k)}\right)}{\prod_{k=t}^{T-1} \mu\left(a^{(k)} \mid s^{(k)}\right)}$.

$$V^{\pi}(s) = \mathbb{E}_{\mu}\left[\frac{\prod_{k=t}^{T-1} \pi\left(a^{(k)} \mid s^{(k)}\right)}{\prod_{k=t}^{T-1} \mu\left(a^{(k)} \mid s^{(k)}\right)} G^{(t)}\right] \tag{18}$$

$$= \mathbb{E}_{\mu}\left[G^{(t)}_{\pi/\mu}\right] \tag{19}$$

The final value-function update can be summarized as

$$\hat{V}(s_t) = \hat{V}\left(s_t\right) + \alpha\left[G_t^{\pi/\mu} - \hat{V}(s_t)\right] \tag{20}$$

3.3 Importance Sampling for Off-policy Temporal Differencing

In this section, we introduce the concept of importance sampling for off-policy TD prediction, and show how to update the value-function update steps, to converge upon the correct value-function.

Importance sampling is a technique that helps us estimate the values of the value-function of the target policy, when given samples from a different a behavioral policy.

We will derive the importance sampling based update equation below. We start by expanding our value-function, based on our pre-defined definitions.

$$V^{\pi}(s) = \mathbb{E}_{\pi}\left[G^{(t)} \mid s^{(t)} = s\right] = \mathbb{E}_{\pi}\left[r^{(t)} + \gamma V^{\pi}\left(s^{(t+1)}\right) \mid s^{(t)} = s\right] \tag{21}$$

$$= \sum_{s^{t+1}, a^{t}} \pi\left(a^{(t)} \mid s^{(t)}\right) p\left(s^{(t+1)} \mid s^{(t)}, a^{(t)}\right)\left(r^{(t)} + \gamma V^{\pi}\left(s^{(t+1)}\right)\right) \tag{22}$$

We can see that our current rollouts are conditional sampled from the target policy, π. Thus, we will introduce a trick that will help us transition the sampling process to one sampled by the behavioral policy, μ. We multiply the numerator and the denominator with the corresponding sampled sequence from the behavioral policy, μ. This does not affect the mathematical integrity of our update, as these values cancel out each other mathematically.

Next, we rearrange the numerator and the denominator to get a sampled sequence that is defined by our behavioral policy.

$$V^{\pi}(s) = \sum_{s^{t+1}, a^{t}} \frac{\mu\left(a^{(t)} \mid s^{(t)}\right) p\left(s^{(t+1)} \mid s^{(t)}, a^{(t)}\right)}{\mu\left(a^{(t)} \mid s^{(t)}\right) p\left(s^{(t+1)} \mid s^{(t)}, a^{(t)}\right)} \pi\left(a^{(t)} \mid s^{(t)}\right) p\left(s^{(t+1)} \mid s^{(t)}, a^{(t)}\right)\left(r^{(t)} + \gamma V^{\pi}\left(s^{(t+1)}\right)\right) \tag{23}$$

$$= \sum_{s^{t+1}, a^{t}} \mu\left(a^{(t)} \mid s^{(t)}\right) p\left(s^{(t+1)} \mid s^{(t)}, a^{(t)}\right) \frac{\pi\left(a^{(t)} \mid s^{(t)}\right)}{\mu\left(a^{(t)} \mid s^{(t)}\right)}\left(r^{(t)} + \gamma V^{\pi}\left(s^{(t+1)}\right)\right) \tag{24}$$

We finally reformulate the equation as an expectation and define a new importance sampling ratio $\frac{\prod_{k=t}^{T-1} \pi\left(a^{(k)} \mid s^{(k)}\right)}{\prod_{k=t}^{T-1} \mu\left(a^{(k)} \mid s^{(k)}\right)}$.

$$V^{\pi}(s) = \mathbb{E}_{\mu}\left[\underbrace{\frac{\pi\left(a^{(t)} \mid s^{(t)}\right)}{\mu\left(a^{(t)} \mid s^{(t)}\right)}}_{\text{Importance weighting}} \overbrace{\left(r^{(t)} + \gamma V^{\pi}\left(s^{(t+1)}\right)\right)}^{\text{One-step TD estimate}}\right] \tag{25}$$

The final value-function update can be summarized as

$$\hat{V}(s_t) = \hat{V}\left(s_t\right) + \alpha\left[\frac{\pi\left(a_t \mid s_t\right)}{\mu\left(a_t \mid s_t\right)}\left[r_t + \gamma \hat{V}\left(s_{t+1}\right)\right] - \hat{V}\left(s_t\right)\right] \tag{26}$$

3.4 Off-policy vs On-Policy Temporal Differencing and MC Prediction

In this section, we present the pseudo-code for the different model-free prediction algorithms presented thus far.

Algorithm 2 MC-Prediction(π, α)

for $e = 1, \cdots, E$ **do**
  $\left\{s^{(t)}, a^{(t)}, r^{(t)}\right\}_{t=0}^{T} \sim \mathcal{E} \mid \pi$ ▷ Sampled from agent's current policy
  for $t = 0, \cdots, T$ **do**
    $G^{(t)} \leftarrow \sum_{i=t}^{T} r^{(t)}$ ▷ Summation of rewards
    $V\left(s^{(t)}\right) \leftarrow V\left(s^{(t)}\right) + \alpha\left(G^{(t)} - V\left(s^{(t)}\right)\right)$ ▷ Value Function Update
  end for
end for
return $V\left(s^{(t)}\right)$

Algorithm 3 TD-Prediction($\pi, \mu, \alpha, s^{(0)}$)

for $t = 0, \cdots, T$ **do**
  $\left\{s^{(t+1)}, a^{(t)}, r^{(t)}\right\} \sim \mathcal{E} \mid \pi, s^{(t)}$ ▷ Sampled from agent's current policy
  $G^{(t)}(1) = \left(r^{(t)} + \gamma V^{\pi}\left(s^{(t+1)}\right)\right)$ ▷ Bootstrapped Value
  $V^{\pi}\left(s^{(t)}\right) \leftarrow V^{\pi}\left(s^{(t)}\right) + \alpha\left(G^{(t)}(1) - V^{\pi}\left(s^{(t)}\right)\right)$ ▷ Value Function Update
end for
return $V\left(s^{(t)}\right)$

Algorithm 4 OffPolicy-MC-Prediction (π, μ, α)

for $e = 1, \cdots, E$ **do**
  $\left\{s^{(t)}, a^{(t)}, r^{(t)}\right\}_{t=0}^{T} \sim \mathcal{E} \mid \mu$ ▷ Sampled from behavioral policy
  for $t = 0, \cdots, T$ **do**
    $G^{(t)} \leftarrow \sum_{i=t}^{T} r^{(t)}$ ▷ Summation of rewards
    $G_{\pi/\mu}^{(t)} = \frac{\prod_{k=t}^{T-1} \pi\left(a^{(k)}|s^{(k)}\right)}{\prod_{k=t}^{T-1} \mu\left(a^{(k)}|s^{(k)}\right)} G^{(t)}$ ▷ Importance sampling
    $V^{\pi}\left(s^{(t)}\right) \leftarrow V^{\pi}\left(s^{(t)}\right) + \alpha\left(G_{\pi/\mu}^{(t)} - V^{\pi}\left(s^{(t)}\right)\right)$ ▷ Value Function Update
  end for
end for
return $V\left(s^{(t)}\right)$

Algorithm 5 OffPolicy-TD-Prediction($\pi, \mu, \alpha, s^{(0)}$)

for $t = 0, \cdots, T$ **do**
  $\left\{s^{(t+1)}, a^{(t)}, r^{(t)}\right\} \sim \mathcal{E} \mid \mu, s^{(t)}$ ▷ Sampled from behavioral policy
  $G_{\pi/\mu}^{(t)}(1) = \frac{\pi\left(a^{(t)}|s^{(t)}\right)}{\mu\left(a^{(t)}|s^{(t)}\right)}\left(r^{(t)} + \gamma V^{\pi}\left(s^{(t+1)}\right)\right)$ ▷ Importance sampling with bootstrapping
  $V^{\pi}\left(s^{(t)}\right) \leftarrow V^{\pi}\left(s^{(t)}\right) + \alpha\left(G_{\pi/\mu}^{(t)}(1) - V^{\pi}\left(s^{(t)}\right)\right)$ ▷ Value Function Update
end for
return $V\left(s^{(t)}\right)$

3.5 Review of Prediction Methods

In this section, we will present a few tables summarizing the concepts presented over the past few lectures.

Algorithms	Return estimate $G^{(t)}$
MC	$G^{(t)} = \sum_{i=t}^{T} r^{(i)}$
TD(0)	$G_0^{(t)} = r^{(t)} + \gamma V\left(s^{(t+1)}\right)$
N-step TD	$G^{(t)}(N) = \sum_{i=t}^{t+N-1} \gamma^{i-t} r^{(i)} + \gamma^N V\left(s^{(t+N)}\right)$
λ-Return	$G_\lambda^{(t)} = (1-\lambda)\sum_{n=1}^{\infty} \lambda^{n-1} G^{(t)}(n)$
TD(λ)	$G_z^{(t)} = z^{(t)}(s)\left[r^{(t)} + \gamma V\left(s^{(t+1)}\right)\right]$
IS-MC	$G_{\pi/\mu}^{(t)} = \prod_{i=t}^{T} \frac{\pi\left(a^{(i)}\middle\vert s^{(i)}\right)}{\mu\left(a^{(i)}\middle\vert s^{(i)}\right)} \sum_{j=t}^{T} r^{(j)}$
IS-TD(0)	$G_{\pi/\mu}^{(t)}(1) = \frac{\pi\left(a^{(t)}\middle\vert s^{(t)}\right)}{\mu\left(a^{(t)}\middle\vert_{s(t)}\right)}\left[r^{(t)} + \gamma V\left(s^{(t+1)}\right)\right]$

Table 2: Review of Prediction Methods

Algorithms	Return estimate $G^{(t)}$
MC	$G^{(t)} = \sum_{i=t}^{T} r^{(i)}$
TD(0)	$G_0^{(t)} = r^{(t)} + \gamma Q\left(s^{(t+1)}, a^{(t+1)}\right)$
N-step TD	$G^{(t)}(N) = \sum_{i=t}^{t+N-1} \gamma^{i-t} r^{(i)} + \gamma^N Q\left(s^{(t+N)}, a^{(t+N)}\right)$
λ-Return	$G_\lambda^{(t)} = (1-\lambda)\sum_{n=1}^{\infty} \lambda^{n-1} G^{(t)}(n)$
TD(λ)	$G_z^{(t)} = z^{(t)}(s,a)\left[r^{(t)} + \gamma Q\left(s^{(t+1)}, a^{(t+1)}\right)\right]$
IS-MC	$G_{\pi/\mu}^{(t)} = \prod_{i=t}^{T} \frac{\pi\left(a^{(i)}\middle\vert s^{(i)}\right)}{\mu\left(a^{(i)}\middle\vert s^{(i)}\right)} \sum_{j=t}^{T} r^{(j)}$
IS-TD(0)	$G_{\pi/\mu}^{(t)}(1) = \frac{\pi\left(a^{(t)}\middle\vert s^{(t)}\right)}{\mu\left(a^{(t)}\middle\vert s^{(t)}\right)}\left[r^{(t)} + \gamma Q\left(s^{(t+1)}, a^{(t+1)}\right)\right]$

Table 3: Q-Value Equivalents

References

[1] R. S. Sutton and A. G. Barto. *Reinforcement learning: an introduction*. Adaptive computation and machine learning. MIT Press, Cambridge, Mass, 1998.